504-REN-XXX

MISSION OPERATIONS AND DATA SYSTEMS DIRECTORATE

Renaissance First Generation Architecture Catalog of Building Blocks (DRAFT 3)

XX June 1995

National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland

Renaissance
First Generation Architecture
Catalog of Building Blocks
(DRAFT 1)

02 February 1995

Prepared Under Contract NAS5-31260
Task Assignment 4-10-201

Approved By:

Ed Seidewitz	Date
Renaissance Team	

Management Approval:

Gary Meyers	Date
Renaissance Team Leader	

Goddard Space Flight Center
Greenbelt, Maryland

Preface

This document was developed by the Renaissance Team Application Engineering Group of the Systems Engineering Office (Code 504), and represents the combined efforts of MO&DSD staff and supporting contractors of the Systems, Engineering, and Analysis Support (SEAS) and the Space Network Systems Engineering contracts. The document was integrated by Stanford Telecommunications Inc. under the Space Network Systems Engineering contract.

This document is under the configuration management of the Renaissance Team. Configuration Change Requests (CCRs) to this document shall be submitted to the Renaissance Team, along with supportive material justifying the proposed change. Changes to this document shall be made by document change notice (DCN) or by complete revision.

Questions and proposed changes concerning this document shall be addressed to:

Ed Seidewitz
Renaissance Application Engineering Group
Code 552.3
Goddard Space Flight Center
Greenbelt, Maryland 20771

Abstract

The **Renaissance First Generation Architecture Catalog of Building Blocks** provides a description of the subsystem, element, and subelement (TBS) building blocks identified by the Renaissance Team for use in developing future ground data systems. This document is intended to support future missions in identifying and selecting components for the design of their ground data systems.

The Renaissance approach is based on enhanced support for mission requirements, with improved productivity, through the elimination of redundant mission support elements and a high level of re-use of application software from mission to mission.

This document describes building blocks (or components) that are applicable across multiple missions. Mission-specific components, providing capabilities unique to only one (or relatively few) flight missions, are not included.

This document first provides an overview of a generic "first generation" Renaissance ground data system architecture to establish the context in which to view the building blocks. The various categories by which the building blocks are classified are discussed next. The remainder of the document presents a summary listing of all building blocks followed by a catalog that provides one-page descriptions for each individual building block. References to associated requirements documents and/or specifications are provided where available. Due to the nature of the building block definition and development process, this catalog will be subject to on-going refinement. Comments, additions, and recommendations for improvement are highly welcome.

Keywords: *ground data systems, Renaissance, spacecraft data processing, Catalog, Building Blocks.*

Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

The Reusable Network Architecture for Interoperable Space Science, Analysis, Navigation, and Control Environments (Renaissance) is a new approach to providing ground data processing systems to support Mission Operations and Data Systems (Code 500) customers in a cost effective, timely manner. This new approach redefines the architecture of the systems developed by Code 500, the way in which these systems are developed, and the role Code 500 will play in the operation of these systems.

The Renaissance approach is based on the concept of using reusable building blocks (which may be hardware, software, and/or firmware) that may be integrated and enhanced to support a single mission or mission series (such as the Small Explorers). These building blocks incorporate workstation/file server/LAN technology, widely accepted industry standards, and (when possible and appropriate) commercial off-the-self (COTS) technology. The design of the building blocks is focused on mission needs rather than facility boundaries.

This document is a product of the Renaissance Team. The Renaissance Team was chartered by Code 500 to specifically meet five goals:

a. Identify the building blocks needed to make the Renaissance approach successful.

b. Identify the standards and define the methods necessary to make the development of these building blocks successful.

c. "Contract" with the Code 500 divisions to implement and maintain the building blocks.

d. Develop a plan for transitioning Code 500 to the new architectural approach.

e. During the transition period, work with mission teams to identify specific architectures for support of near-term missions using the new building blocks whenever feasible.

1.2 Document Scope and Organization

The purpose of this document is to provide a catalog of Renaissance building blocks for use in MO&DSD ground data systems. The catalog consists of descriptions of generic building blocks within the context of the generic Renaissance architecture. Most of the building blocks are currently available in some, though many are currently undergoing evolution to provide the complete capabilities described in this catalog. There are also some planned building blocks that are still under development.

There are two main uses for this catalog. First, it provides the definitive list of Renaissance building block definitions to be allocated to various MO&DSD "centers of expertise" for implementation. Second, this list also provides a reference of existing (and planned) Renaissance building blocks available to support a mission. In addition, the document provides a general context of how these building blocks fit together to create a complete mission ground data system architecture.

The organization of this document is as follows:

- The remainder of this section provides an overview of the Renaissance generic ground data system (GDS) architecture and the context of the building blocks within this architecture.
- Section 2 provides descriptions of the various real-time, off-line, and foundation subsystems that make up the "first generation" Renaissance architecture. This section provides a framework for organizing the Renaissance building blocks and understanding their use and interaction.
- Section 3 provides a summary description of the building blocks and the actual Renaissance building block catalog. The summary list and catalog are organized according to the subsystems defined in Section 2. The catalog contains a description page for each of the Renaissance building blocks. Each description page provides detailed information including a functional description, performance characteristics, key inputs and outputs, relationship to other building blocks, the hardware environment, the programming language, disk space requirements, etc.

1.3 Generic Ground Data System Architecture Concept

Figure 1-1 gives an overview of a satellite ground data system in the Renaissance context. Actual communications with the satellite is via one or more ground stations. In this context, a "ground station" may include a Ground Network station, the White Sands ground terminals for the space network, the Jet Propulsion Laboratory Deep Space Network facility, or any other tracking and data relay network facility from which a mission may receive data.

Data is transported to and from the ground stations via the "closed" (i.e., secure) MODNet network. MODNet is a NASA Communications (NASCOM) supported network based on the commercial Internet protocol (IP). The Mission Operations Center (MOC), the Flight Dynamics Facility (FDF), and the Integration and Test (I&T) facility are all within the secure perimeter of the closed MODNet. Security gateways and firewalls provide access from the closed MODNet to the "open" (unsecured) MODNet and the Internet. The unsecure networks allow connectivity to the Science Operations Center (SOC) (though this could be placed within the closed MODNet, particularly if it is located at Goddard).

Of the facilities shown in Figure 1-1, only the FDF is necessarily located at Goddard. All other facilities may be geographically located any place network connectivity allows. Indeed, it is even possible to further distribute some of the lower-level functionality shown in Figure 1-1. For example, level-zero processing could easily be moved from the MOC to the SOC, close to the primary cutomer of level-zero products. Indeed, for some missions it may be desirable to completely combine the MOC and SOC operational functions into a single facility located at the

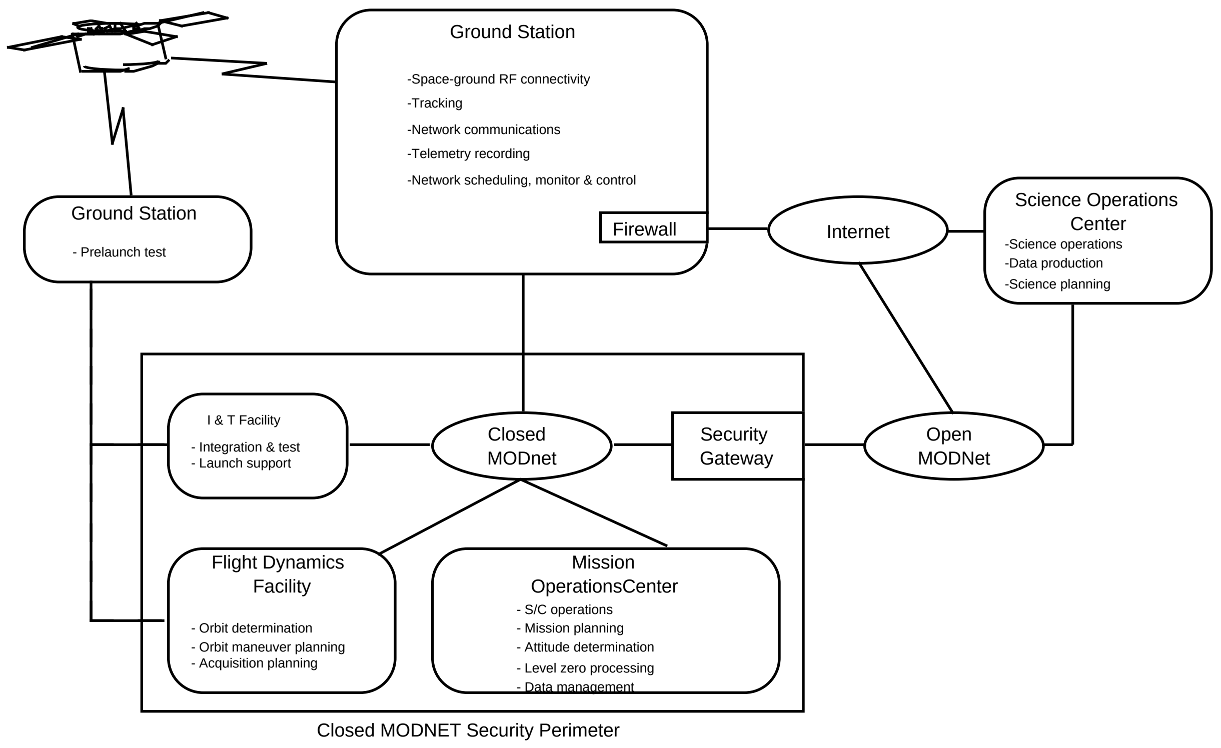


Figure 1-1. Mission Ground Data System Context

invetigators' site. Ultimately, the orbit-related functions now performed in the FDF will also be similarly distributable.

The FDF, ground station, and communication facilities are provided institutionally by Code 500. All other facilities—the MOC, the SOC, and the I&T facility—are developed and tailored specifically for a mission and operated by mission-dedicated teams. This mission-tailored GDS approach allows maximum responsiveness to a specific mission's needs. The approach is made cost effective by integrating the mission-tailored facilities from available Renaissance building blocks.

The generic Renaissance GDS architecture is based on the principles of frequent building block reuse and configuration flexibility to achieve cost efficiency. The Renaissance approach seeks to achieve a high level of reuse among multiple mission GDS implementations by providing a large suite of building blocks. This large complement of building blocks also allows the flexibility needed in constructing GDSs with specific mission requirements. The concept, at its simplest, is analogous to implementing a computer hardware configuration using a pre-existing set of IC boards, which plug into the system backplane (common among all implementations), and the selection of specific IC chips to customize some of these boards. Renaissance building block re-use is based on an architecture concept of software system construction from a family of interchangeable software components.

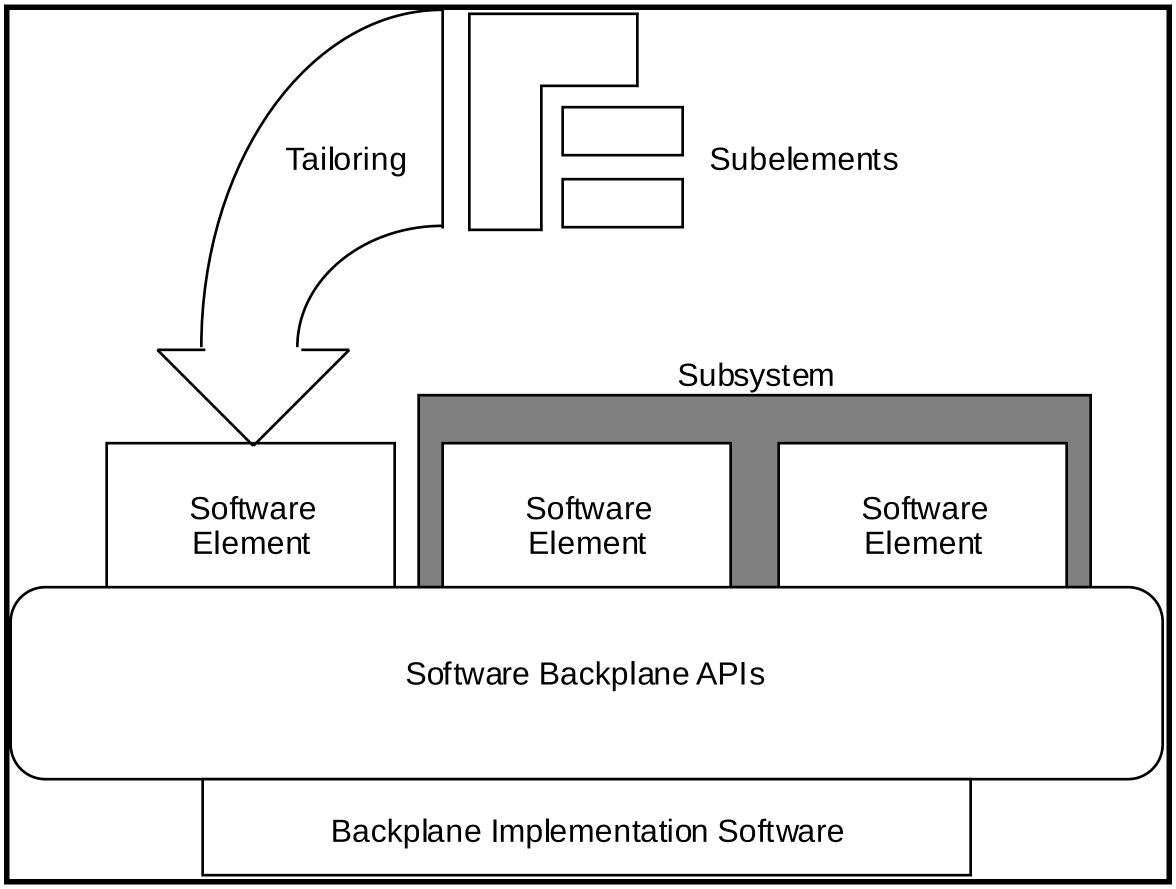


Figure 1-2. The Software Backplane

The Renaissance GDS software is partitioned into subsystems, elements, and subelements. A subsystem comprises those software elements which together perform a high level service for the system, e.g., spacecraft operations monitoring. Software elements are those software components that communicate with other elements only through a "software backplane" (see Figure 1-2). The software backplane provides a common set of distributed communication and control mechanisms via a suite of standard application programming interfaces (APIs).

A software element is constructed from subelements that may be linked together or may be separate tasks or processes. A subelement is defined to be a software component which requires separate specification to support flexibility of configuration for support of a specific mission. The subelements of a single element are generally expected to all reside on a single platform. Subelements within a software element may communicate among themselves by means other than the software backplane.

Those software components which provide the functionality of the software backplane are considered to be separate from the software elements. These software components will generally include the mission specific operating system and support utilities which are provided for the mission-selected platforms and networks. Note that while the backplane components themselves may change from one mission implementation to the next, the standard backplane APIs do not.

The generic architectural concepts cataloged here are more fully described in the *Renaissance Generic Architecture* document (reference TBD). The building blocks discussed in the present building block catalog are primarily intended for use within the context of the MOC, though they can also provide the basis for at least some of the capabilities of the other facilities. Further guidance on the use and tailoring of building blocks within the Renaissance generic architecture can be found in the *Renaissance Configuration Guide* (reference TBD).

2. First Generation Architecture

In order to remain competitive with other emerging alternatives, the MO&DSD needs to move towards an integrated GDS approach not just in the long term, but also for near-to-medium term missions This includes mission such as ACE, TRACE and Landsat-7, which the Directorate is already committed to support. Cost-effectively supporting such near-term missions requires Renaissance to leverage off existing ground data system legacy.

In order to deal with this near-term problem, the Renaissance Team has defined a "first generation" architecture that evolves from current real-time and off-line data system approaches. This allows the adaptation of legacy software while moving to state-of-the-practice technology and standards. Unfortunately, the MO&DSD currently has disparate real-time and off-line legacies that are not easily integrated. Thus the first generation architecture will not achieve the full integration and interoperability that is the goal of Renaissance. Nevertheless, it will provide a competitive near-term solution while work continues on a state-of-the-art "second generation" approach.

Figure 2-1 diagrams the first-generation Renaissance generic architecture at a high level (see also the *Renaissance Generic Architecture* document (reference TBD). As shown, it is divided into three major segments:

- *Real-time software* that deals with the command, control and monitoring of spacecraft. The architecture of this software is based on the legacy Transportable Payload Operations Control Center (TPOCC) architecture (Zhu, 1994) and focuses on the real-time flow of data and commands.

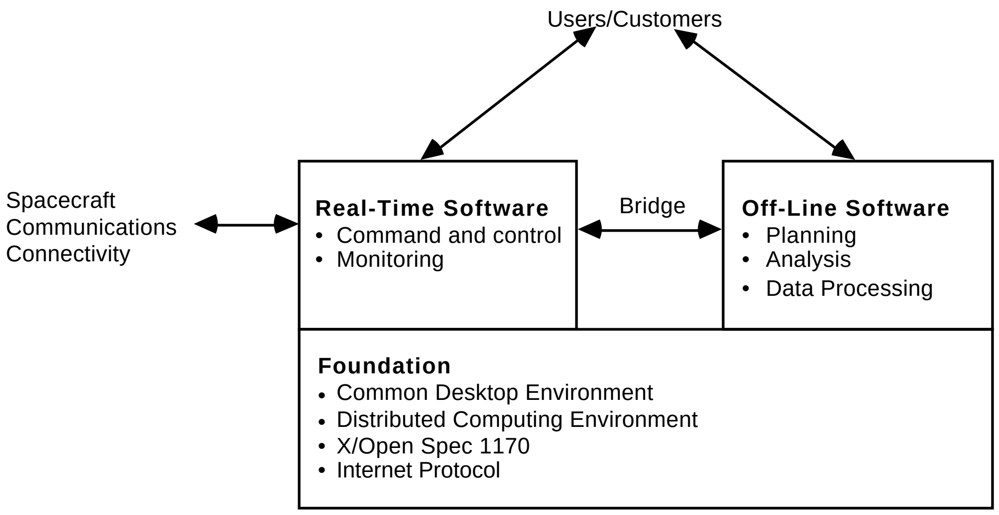


Figure 2-1 First Generation Architecture Overview

- *Off-line software* that deals with mission planning, analysis and data processing, including delivery of products to external customers. The architecture of this software is based primarily on work done for the Flight Dynamics Distributed System (FDDS) (Green, 1994), the Packet Processor II (Pacor II) system (Principe, 1993) and mission planning and command management systems. It focuses on the long-term management and processing of stored data.

- A common *foundation* underlies both the real-time and off-line software. This foundation is composed of COTS software that conforms with industry standards such as the Common Desktop Environment (CDE), the Distributed Computing Environment (DCE) and the X/Open Spec 11/70 (see the *Renaissance Standards* document for a complete discussion of the standards adopted for the foundation (Stottlemyer, 1994).

The software backplane approach discussed in Section 1.3 is used in each of the real-time and off-line segments. However, there is not a common backplane across all real-time and off-line software. Instead, there is a restricted “bridge” of interfaces between the real-time and off-line software consisting primarily of shared data files.

The remainder of this document deals only with the first generation Renaissance architecture. This section gives an overview of each of the subsystems in each of the segments of this architecture. The next section provides detailed descriptions of each of the elements shown here. The elements included here are primarily “software elements” in the sense discussed in Section 1.3. However, there are a few instances in which a hardware/firmware solution provides an alternative or a replacement for a software solution (e.g., level-zero processing and frame gateway hardware). For completeness, these hardware building blocks are included here in appropriate subsystems with related software elements.

The following subsections include an "architectural diagram" for each Renaissance subsystem. These diagrams are included to show the context in which each element operates and the interconnections among the elements. Figure 2-2 shows the meaning of the symbols used in these diagrams.

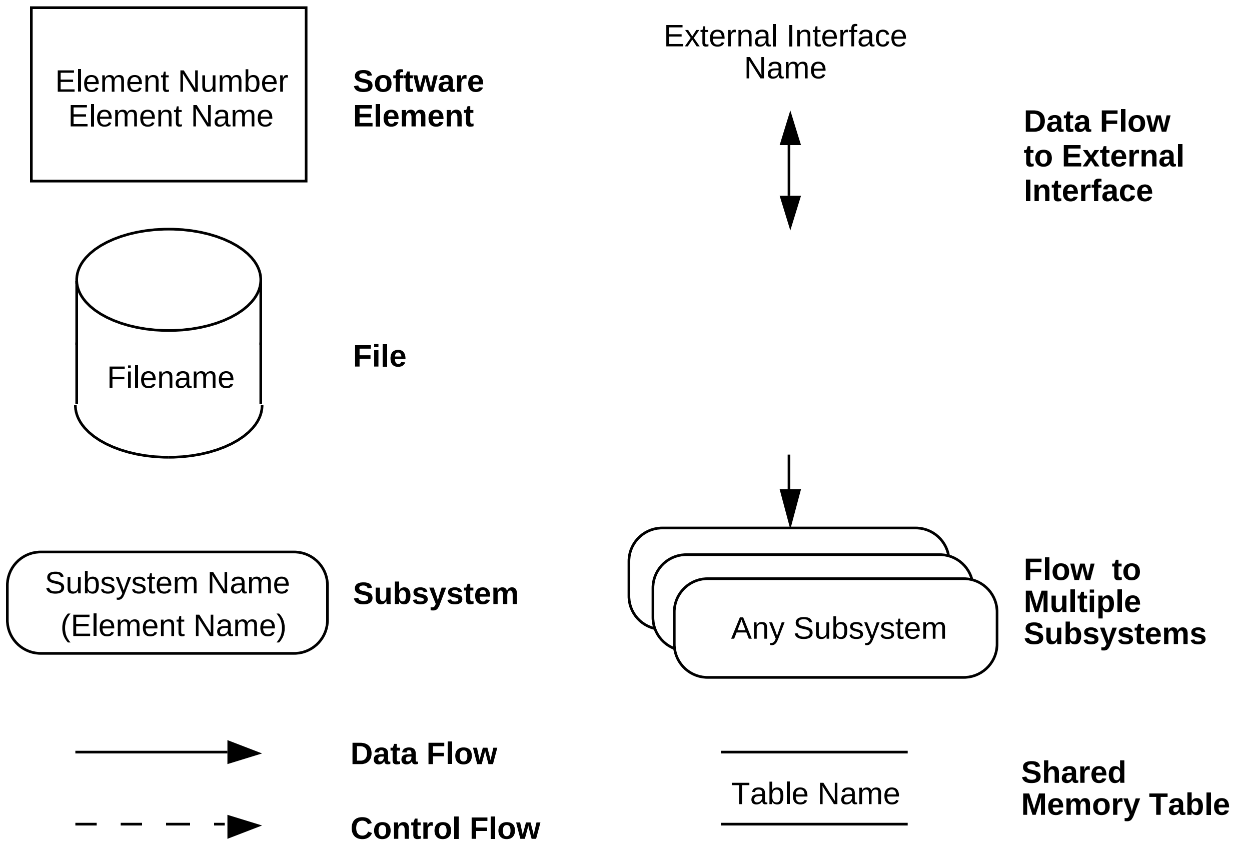


Figure 2-2. Standard Architecture Diagram Symbols

2.1 Real-Time Subsystems

The real-time software elements execute during spacecraft contacts to monitor the health and safety of the spacecraft and to perform command and control functions. The real-time software is divided into seven subsystems, as described in the following paragraphs. Each of the software elements in the real-time architecture communicate using one or more of the real-time software backplane application programming interfaces (APIs). These APIs have been principally drawn from the TPOCC architecture and include the following types of interfaces:

- shared memory (shmem_util)
- data server (libds)
- packet (libpkt)
- external interface (libtnif)
- STOL/State Manager (UTL_StmgrIf)
- event logging (events_util)

In each case the name of the current TPOCC API library has been shown in parentheses. Details on these interfaces may be found in TPOCC documentation (Schwarz, 1993; Zhu, 1994).

In addition, real-time elements may read and write files that are generated and/or used by off-line elements. These files provide the “bridge” between the real-time and off-line software and are shown on the architecture diagrams below for each subsystem. There may also be set-up, configuration or other internal files that are used by the various real-time elements. To avoid clutter, these are considered to be part of the elements themselves and are not shown on the diagrams.

2.1.1 Real-Time Command

The Real-Time Command subsystem transmits real-time commands and command loads to the spacecraft and verifies that these commands have been received successfully. The Real-Time commanding element accesses the command ODB generated from the Off-Line Data Management and Distribution subsystem and receives command loads from the Off-Line Mission Planning subsystem. It also updates the ground reference image (GRI) database for use by the Off-Line State Determination and Validation subsystem.

Commands are sent out of the control center via the External Network Interface (ENIF) which passes a copy of each outgoing block to the Command Echo Processor. This element then compares the outgoing commands against the echoes received from ENIF and reports any discrepancies.

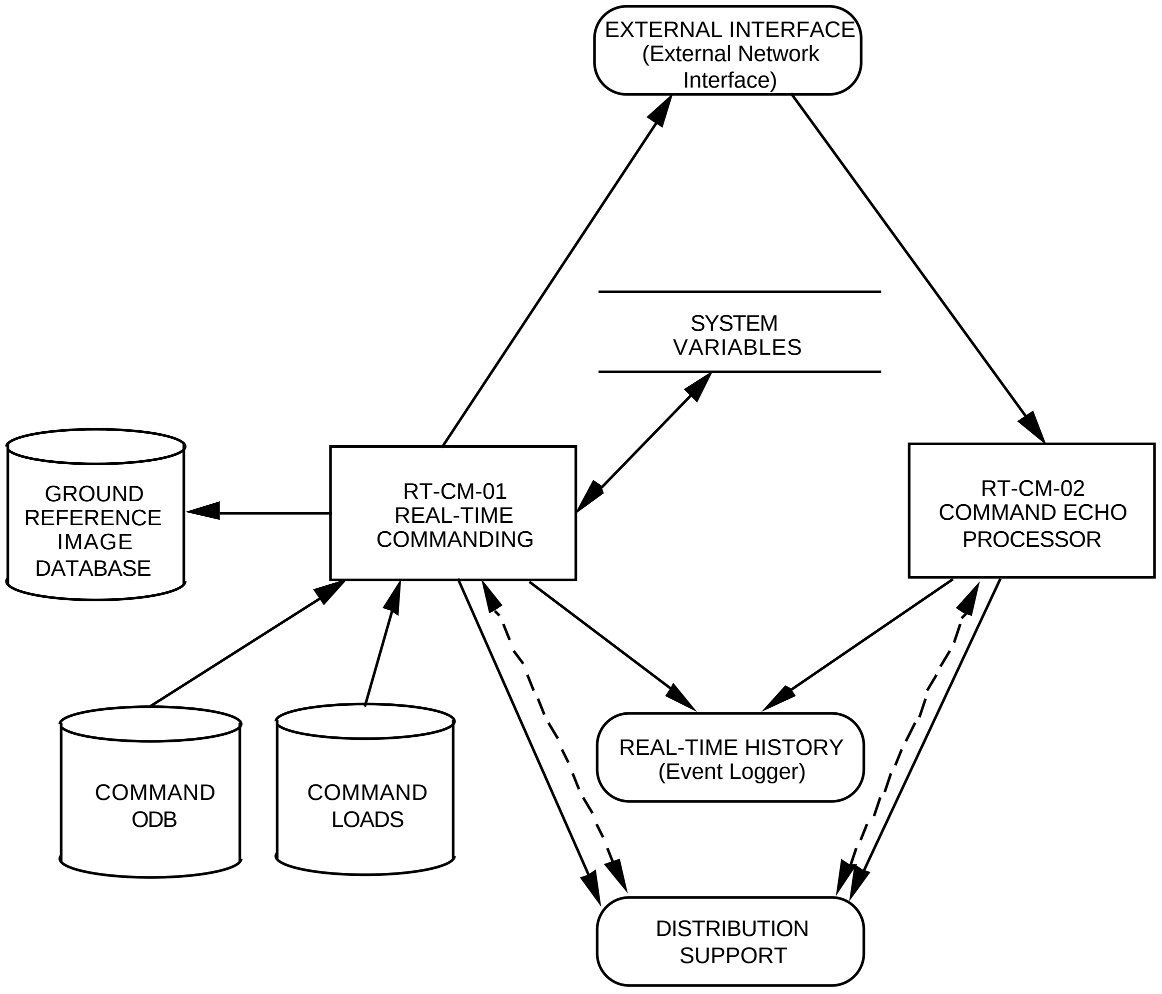


Figure 2.1-1 Real-Time Command

2.1.2 Real-Time Distribution Support

The Real-Time Distribution Support subsystem supports the distributed processing real-time architecture by transferring data and control messages to interested software elements executing on any network host. The Data Server element allows elements on remote hosts to gain access to information stored in the system variable shared memory table and event-driven data. The State Manager maintains overall control of the real-time system by only allowing user actions that are permissible given the current state of the ground system. The software backplane APIs for the Data Server and the State Manager are provided by the libds and UTL_StmgrIf libraries, respectively.

NOTE

Some software elements inherited from the TPOCC front-end architecture currently directly access the system variable shared memory table rather than using the Data Server. The Renaissance architecture does not specifically call for a dedicated front end, but obviously all software elements that directly access the system variable shared memory must reside on the same platform. It is intended to eventually evolve away from any access to system variables other than through the Data Server, allowing greater flexibility of software distribution. The current software backplane APIs for accessing the system variable table are provided by the shmem_util library.

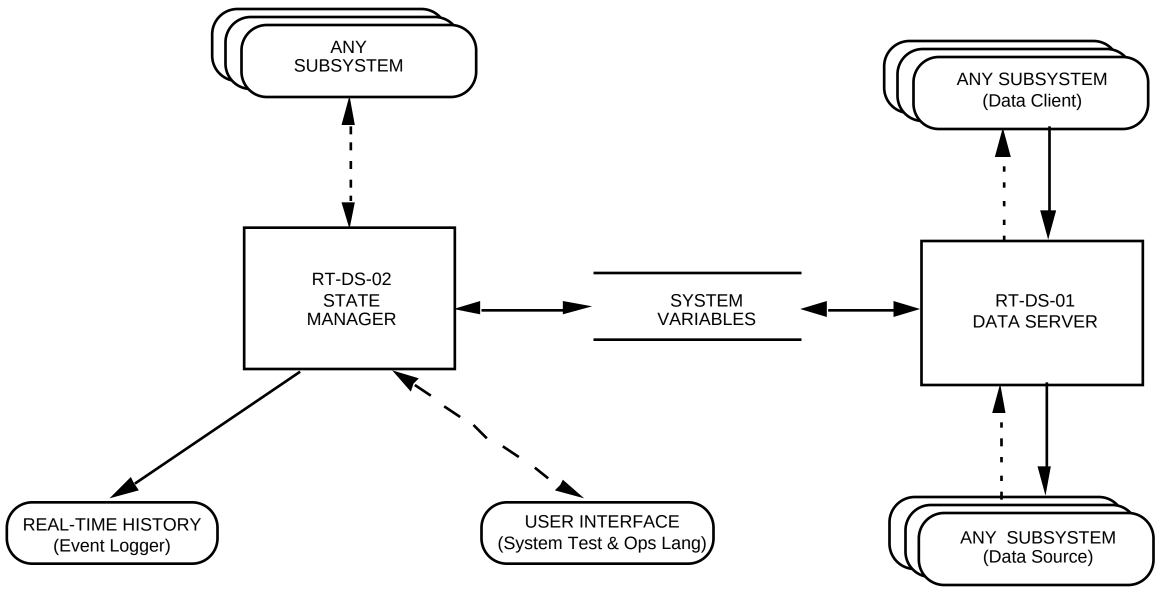

ANY SUBSYSTEM
ANY SUBSYSTEM (Data Client)
RT-DS-02 STATE MANAGER
SYSTEM VARIABLES
RT-DS-01 DATA SERVER
REAL-TIME HISTORY (Event Logger)
USER INTERFACE (System Test & Ops Lang)
ANY SUBSYSTEM (Data Source)

Figure 2.1-2 Real-Time Distribution Support

2.1.3 Real-Time External Interface

The Real-Time External Interface subsystem provides communication services between the ground data system and external elements such as the spacecraft, the tracking and data network ground stations, and the Network Control Center (NCC). Two gateway elements are provided that convert serial bit streams into the IP protocol used internally by the Renaissance architecture. The Frame Gateway element for telemetry data is preferentially located at the ground station rather than in the MOC, so that all wide-area communication is via the IP protocol (as discussed in Section 1.3). The frame gateway may also be used in an integration and test facility to provide direct connectivity with the spacecraft on the ground. It is also possible to locate a frame gateway in the MOC so that telemetry may be received using the older Nascom serial protocol. The Nascom to TCP/IP Gateway element provides similar connectivity to the old Nascom network for non-telemetry data.

The External Network Interface (ENIF) element receives incoming messages using the IP protocol and filters them to the appropriate element for further processing. These additional functions include calculating frame statistics, handling embedded frame synchronization, extracting and serving packets, and providing communications support for various types of networks (NCC, DSN, etc.). The ENIF is also used as a standard interface for sending real-time data (such as commands) out of the MOC over external networks. The software backplane APIs for ENIF are provided by the libtnif library. The APIs for the Packet Extractor/Server are provided by the libpkt library.

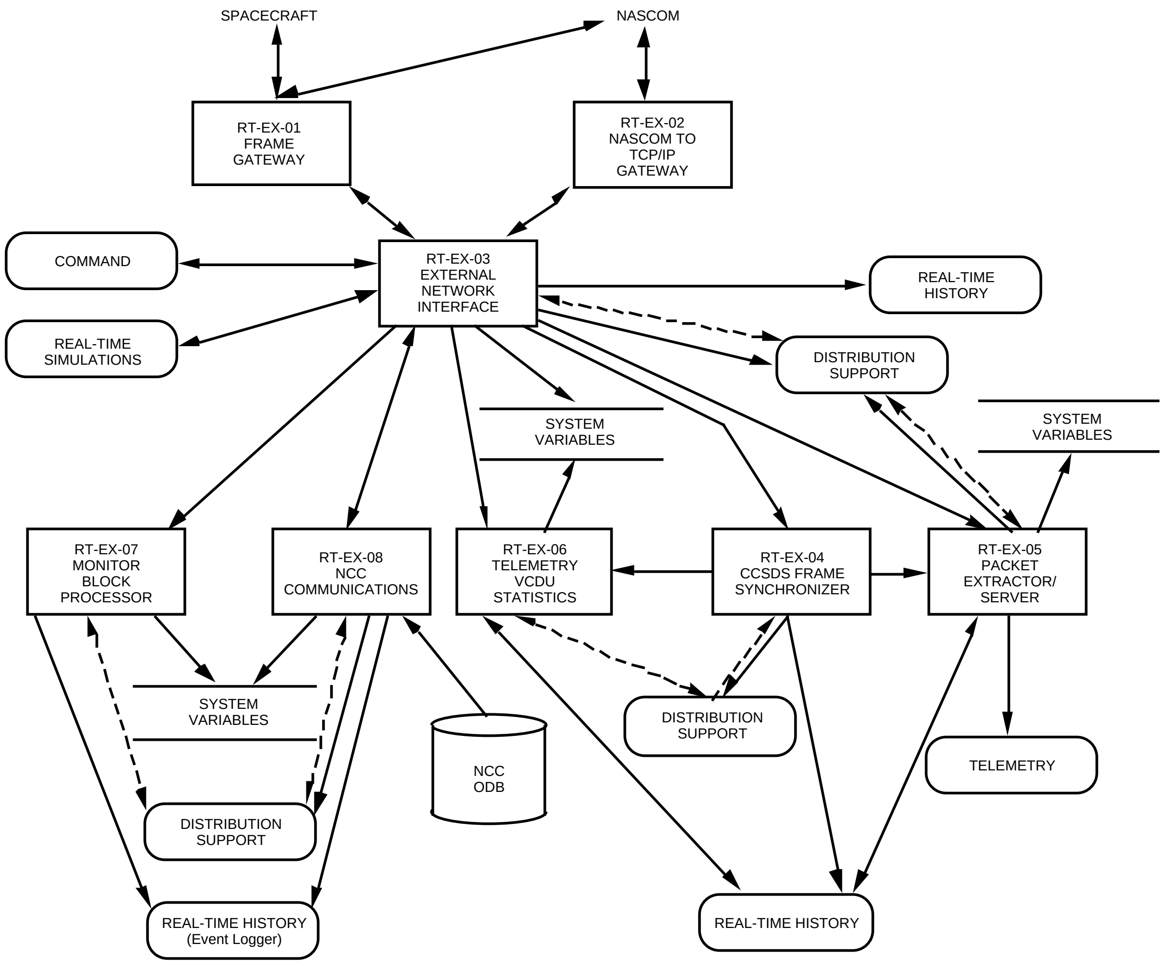

SPACECRAFT
NASCOM
RT-EX-01 FRAME GATEWAY
RT-EX-02 NASCOM TO TCP/IP GATEWAY
COMMAND
RT-EX-03 EXTERNAL NETWORK INTERFACE
REAL-TIME HISTORY
REAL-TIME SIMULATIONS
DISTRIBUTION SUPPORT
SYSTEM VARIABLES
SYSTEM VARIABLES
RT-EX-07 MONITOR BLOCK PROCESSOR
RT-EX-08 NCC COMMUNICATIONS
RT-EX-06 TELEMETRY VCDU STATISTICS
RT-EX-04 CCSDS FRAME SYNCHRONIZER
RT-EX-05 PACKET EXTRACTOR/ SERVER
SYSTEM VARIABLES
NCC ODB
DISTRIBUTION SUPPORT
TELEMETRY
DISTRIBUTION SUPPORT
REAL-TIME HISTORY (Event Logger)
REAL-TIME HISTORY

Figure 2.1-3 Real-Time External Interface

2.1.4 Real-Time History

The Real-Time History subsystem captures all real-time data coming into or out of the ground data system and logs it to disk for later review and/or reprocessing. The History Logger element can log telemetry data at both the frame and packet level. The History Replay element may then be used to replay logged telemetry through the Real-Time Telemetry subsystem and/or the External Interface subsystem and, via this, the rest of the real-time software. The history data may also be viewed and analyzed using the Off-Line Data Production and Analysis subsystem.

The Event Logger logs all real-time system event messages for later viewing using the Off-Line Data Production and Analysis subsystem. The software backplane APIs for the event logger are provided by the events_util library.

NOTE

The current TPOCC "history processor" provides both history delogging and replay. However, delogging is properly an off-line activity and thus more logically included in the off-line segment. Further, this approach allows the traditional TPOCC history delogging to be expanded to also include level-zero data analysis functions (see Section 2.2.2 on the Off-Line Data Production and Analysis subsystem).

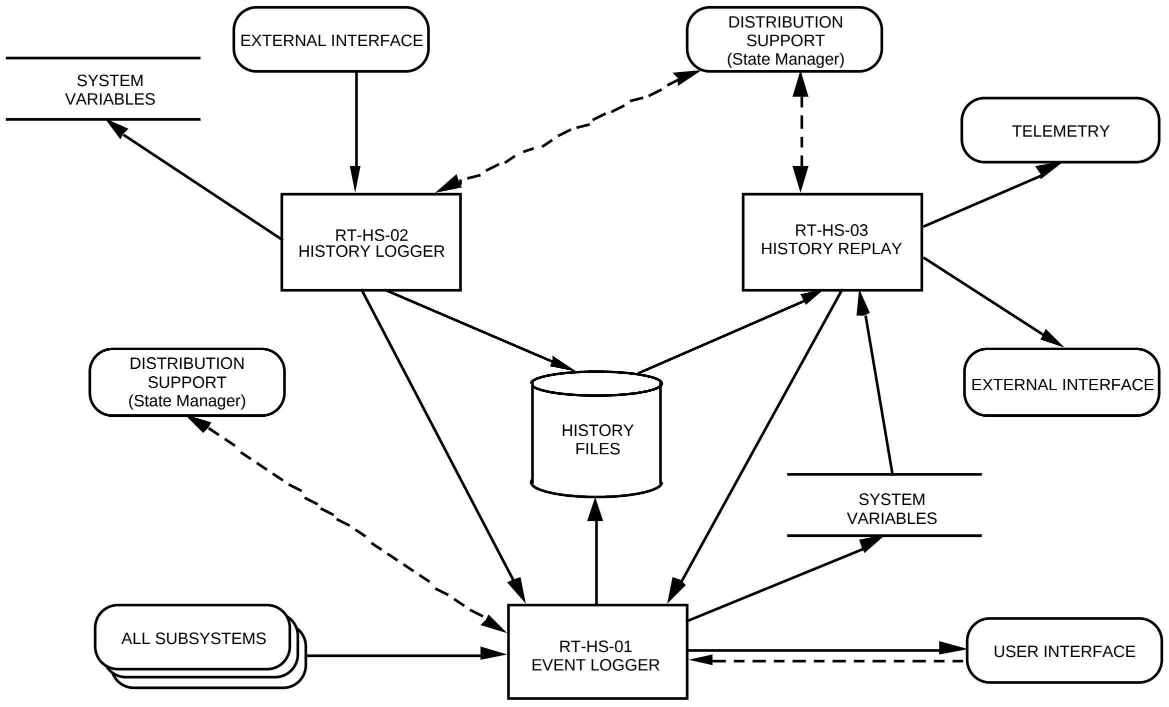


Figure 2.1-4 Real-Time History

2.1.5 Real-Time Simulation

The Real-Time Simulation subsystem provides simulation support to test the real-time ground data system. These software elements generate appropriate telemetry streams for a given mission and accept commands from the ground system. The high-fidelity Spacecraft Simulator also employs spacecraft models to realistically alter telemetry based on command receipt. Both types of simulators draw on information originating in the project data base (via the Off-Line Data Management and Distribution subsystem) to accurately simulate the behavior of a particular spacecraft.

NOTE

The current TPOCC Advanced Spacecraft Simulator (TASS) is the closest legacy simulator at this time to the architecture of the Real-Time Simulation subsystem. TASS needs to only switch its local history subsystem for the generic version to realize this architecture. However, despite its name, TASS only provides telemetry simulation functionality, not full spacecraft simulation. There has already been some work, though, in integrating high-fidelity spacecraft simulators into the TPOCC environment which would be compatible with the architecture of this subsystem.

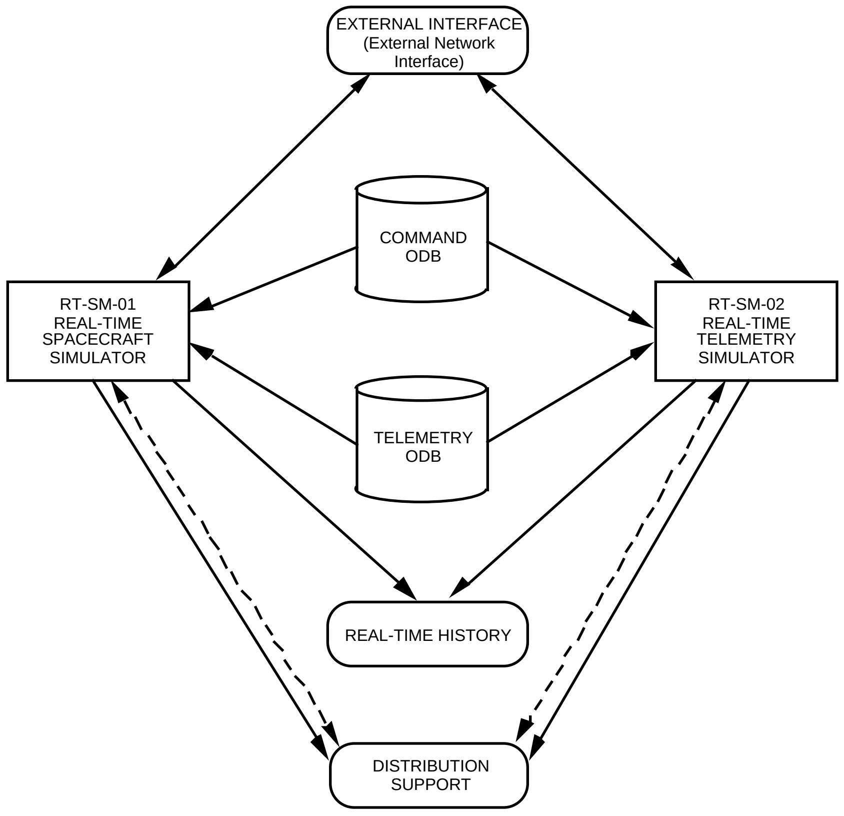

EXTERNAL INTERFACE (External Network Interface)
COMMAND ODB
RT-SM-01 REAL-TIME SPACECRAFT SIMULATOR
RT-SM-02 REAL-TIME TELEMETRY SIMULATOR
TELEMETRY ODB
REAL-TIME HISTORY
DISTRIBUTION SUPPORT

Figure 2.1-5 Real-Time Simulation

2.1.6 Real-Time Telemetry

The Real-Time Telemetry subsystem processes telemetry data received from the spacecraft. This data is received by this subsystem from either the History Replay element or the Packet Extractor/Server. In turn, the telemetry software elements produce raw and engineering unit converted telemetry parameters and dump image files. Sets of telemetry parameters are passed on to other elements within this subsystem that check the spacecraft configuration, compute derived telemetry parameters according to predefined equations, perform real-time attitude determination, and generate telemetry subset files. Level-zero data sets are produced either by the high-speed LZP element in this subsystem or the off-line LZP software in the Off-Line Data Production and Analysis subsystem. Attitude measurement and estimation data files and dump image files are created by this subsystem and passed to the Off-Line State Determination and Validation subsystem. Subset files may be used both for trend analysis in the Off-Line Data Production and Analysis subsystem and as input to Off-Line State Determination and Validation.

NOTE

The Hardware Level-Zero Processor is intended for use with very high telemetry data rates. It is used as a replacement for, rather than in conjunction with, the software Level-Zero Processor element. (Of course, different facility systems may use different LZP solutions, e.g., software LZP in the MOC but hardware LZP in the SOC.) Note that the hardware element receives raw data directly from the frame gateway (possibly over a wide-area network), bypassing both the ENIF and the packet extractor/server.

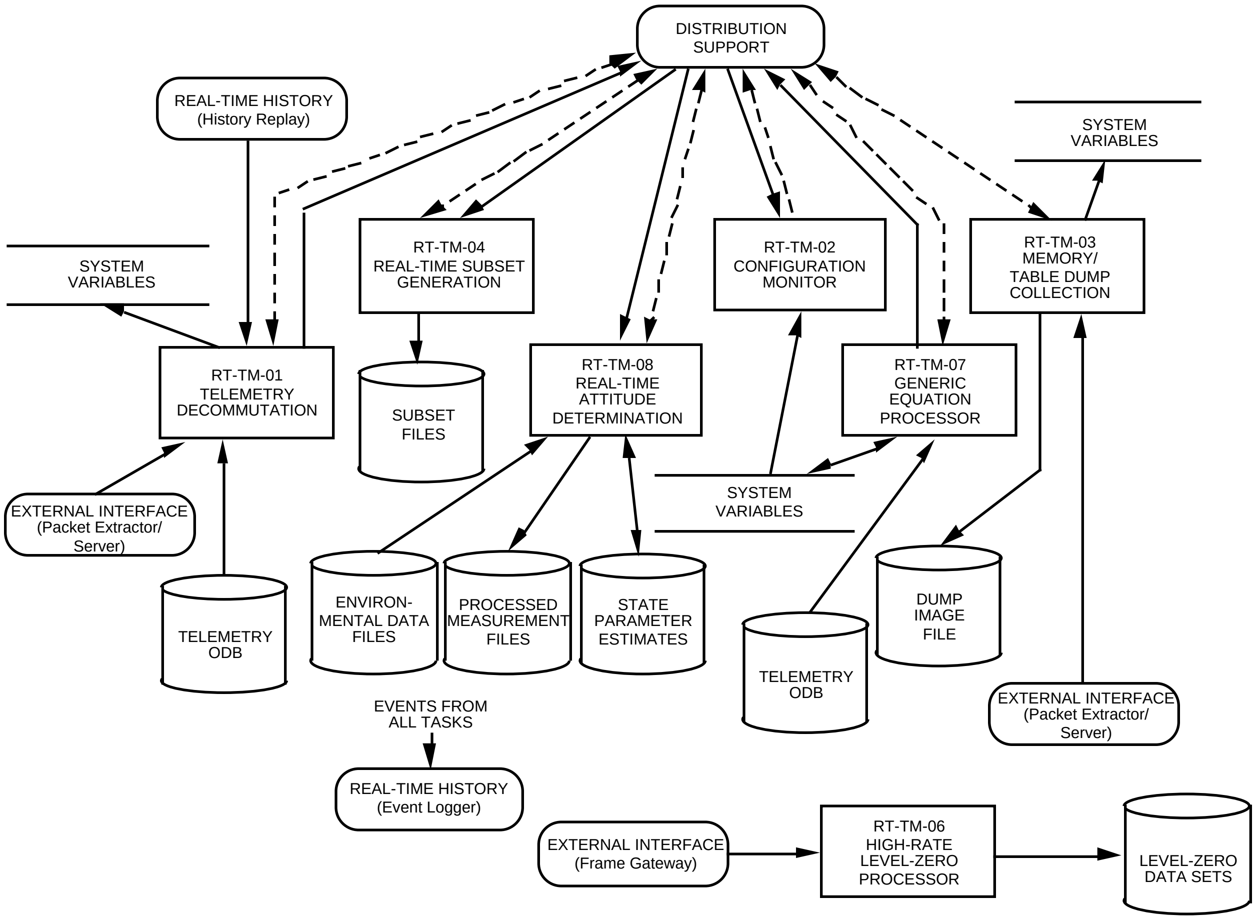

DISTRIBUTION SUPPORT
REAL-TIME HISTORY (History Replay)
SYSTEM VARIABLES
RT-TM-04 REAL-TIME SUBSET GENERATION
RT-TM-02 CONFIGURATION MONITOR
SYSTEM VARIABLES
RT-TM-03 MEMORY/ TABLE DUMP COLLECTION
RT-TM-01 TELEMETRY DECOMMUTATION
SUBSET FILES
RT-TM-08 REAL-TIME ATTITUDE DETERMINATION
RT-TM-07 GENERIC EQUATION PROCESSOR
SYSTEM VARIABLES
EXTERNAL INTERFACE (Packet Extractor/ Server)
TELEMETRY ODB
ENVIRON-MENTAL DATA FILES
PROCESSED MEASUREMENT FILES
STATE PARAMETER ESTIMATES
DUMP IMAGE FILE
TELEMETRY ODB
EXTERNAL INTERFACE (Packet Extractor/ Server)
EVENTS FROM ALL TASKS
REAL-TIME HISTORY (Event Logger)
EXTERNAL INTERFACE (Frame Gateway)
RT-TM-06 HIGH-RATE LEVEL-ZERO PROCESSOR
LEVEL-ZERO DATA SETS

Figure 2.1-6 Real-Time Telemetry

2.1.7 Real-Time User Interface

The Real-Time User Interface subsystem provides the user interface to monitor and control the other real-time subsystems. This subsystem requests data from the data server to produce both graphical displays and ASCII reports. These displays can include text and two-dimensional graphics (i.e., using the Real-Time Display and Generic Spacecraft Analyst Assistant elements) or three-dimensional displays of the spacecraft's orbital environment (using the Space Camera element). The subsystem also includes the System Test and Operations Language element that is used to syntactically check directives and to script operator activity. Finally, the subsystem provides support for listing event messages to a dedicated line printer.

NOTE

The Real-Time Display, Generic Spacecraft Analyst Assistant (GenSAA), and Space Camera elements are based on independently developed legacy user interface systems. Eventually, it would be desirable to combine these separate user interfaces into a single, consistent display element. In addition, the GenSAA element includes an expert system inference engine that could be better used as a separate, stand-alone element or combined with the generic equation processing element in the Real-Time Telemetry subsystem.

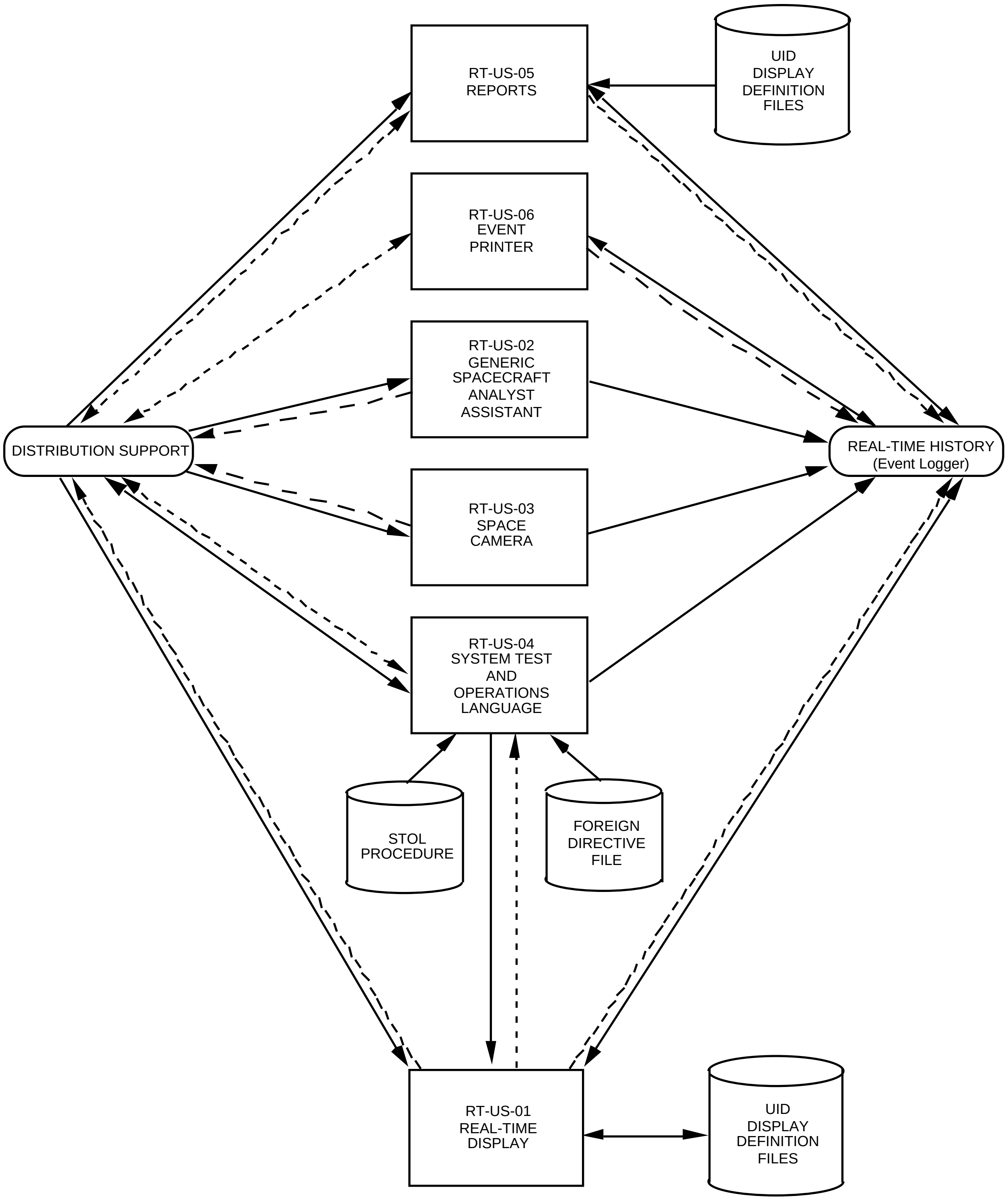


Figure 2.1-7 Real-Time User Interface

2.2 Off-Line Subsystems

The off-line software elements are typically operated between spacecraft contacts to perform system housekeeping functions (such as the distribution of data produced by the MOC) and to assist in the identification and planning of future mission events (though there is no restriction on their use during a spacecraft contact other than availability of resources). The off-line software is divided into six subsystems, as described in the following paragraphs. Each of the software elements in the off-line architecture communicate using stored data that is accessed via the off-line software backplane application program interfaces (APIs). These APIs are based on distributed SQL (for relational database access) and POSIX-compliant system calls (for flat file access). The subsystem architecture diagrams below show all stored data files used for communication between off-line elements, or as bridges to real-time software. There may also be set-up, configuration or other internal files that are used by the various off-line elements. To avoid clutter, these are considered to be part of the elements themselves and are not shown on the diagrams.

2.2.1 Off-Line Data Management and Distribution

The Off-Line Data Management and Distribution subsystem supports the distribution of data to external users, the management of internal data, and the initiation of data processing activities. The Data Distribution element delivers datasets to external users according to predefined distribution specifications. The File Services element manages the backup, archival, and retrieval of datasets according to predefined data management specifications. The General File Formatter element reformats files for use by other applications. The Data Management User Interface updates the data management specifications used by the Data Distribution, File Services and General File Formatter elements. The Processing Initiation element is responsible for scheduling and executing data processing tasks according to a schedule that is a combination of tasks that are started on an absolute time basis and others that are started based on either data availability and/or prior task completion. In addition to the generic management and distribution of data, this subsystem includes the ODB Generation element which generates the operations databases (used by a number of real-time elements) from the project database. (The creation and management of the project database is a mission-specific function and is not shown here).

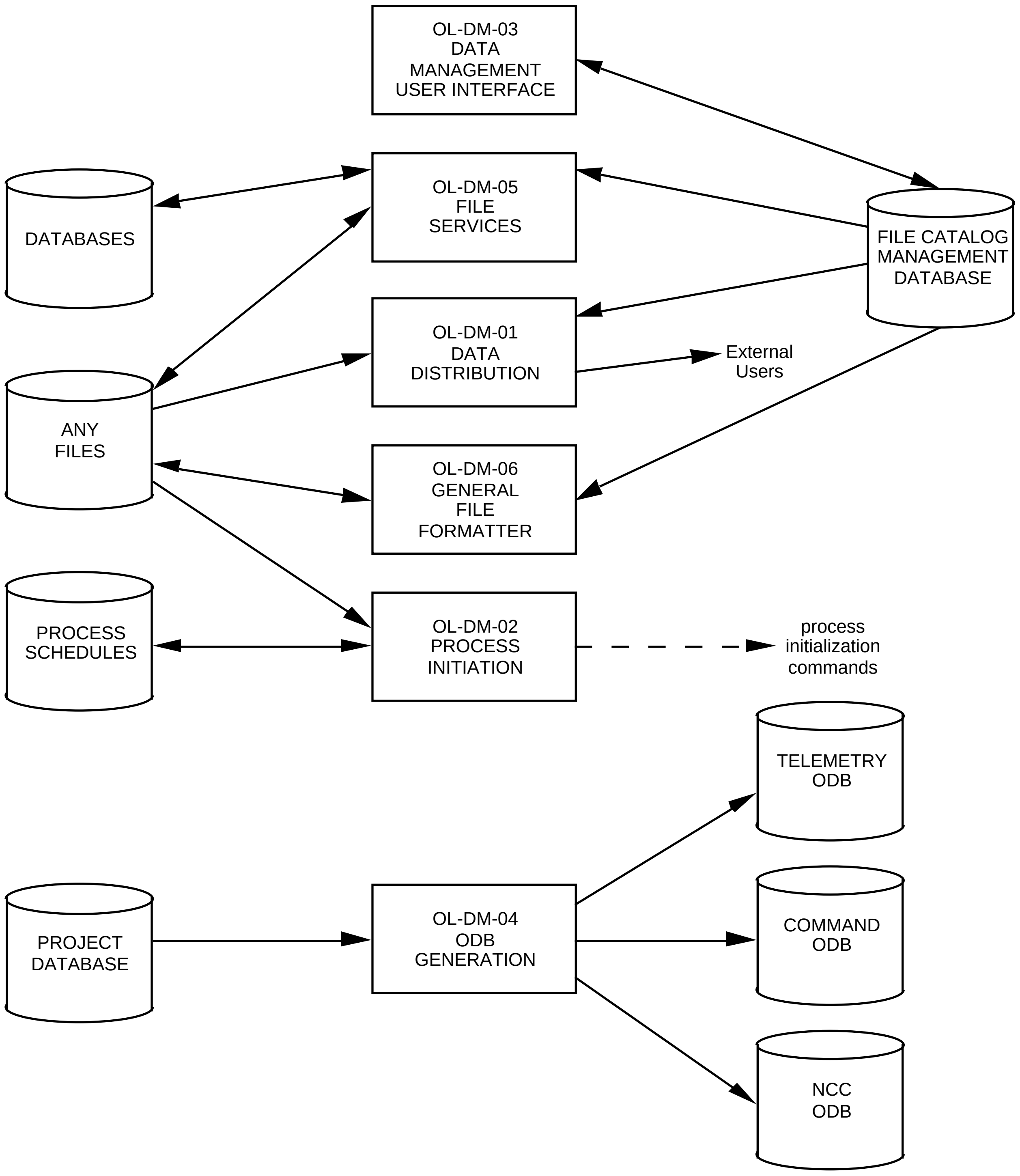


Figure 2.2-1 Off-Line Data Management And Distribution

2.2.2 Off-Line Data Production and Analysis

The Off-Line Data Production and Analysis subsystem supports the activities necessary to process and analyze data. This subsystem provides the ability to generate level zero processed datasets,via the Level Zero Processor, browse information stored in the history files (produced by the Real-Time History subsystem) and level-zero data files via the Events Browser and the Data Browser and Editor. The Data Browser and Editor also provides the ability to edit telemetry data to remove errors in order to successfully level-zero process the data. The LZP User Interface element provides the ability to setup, monitor, and control the Level Zero Processor. The Trend Analysis element provides a general capability to analyze telemetry subset data generated either in real-time (in the Real-Time Telemetry subsystem) or off-line from level-zero processed data (by the High Speed Off-Line Subset element). The subset files produced are also used in other off-line subsystems. Finally, for missions that use time-division multiplexed (TDM) data (such as ACE) the TDM Processor provides the functionality to extract TDM major and minor frame statistics from the level-zero processed data sets and generate reports based on this information.

NOTE

The Data Browser and Editor element combines the functionality of history delogging from TPOCC with quality analysis from Pacor II. This is consistent with a view of telemetry “history” as including data from all phases of processing: frame, packet and level-zero. The functions performed by the Data Browser and Editor are similar for all levels of data. Note also that events data is included in the definition of the history files here. The Events Browser is intended to be a more sophisticated version of the TPOCC event delogger.

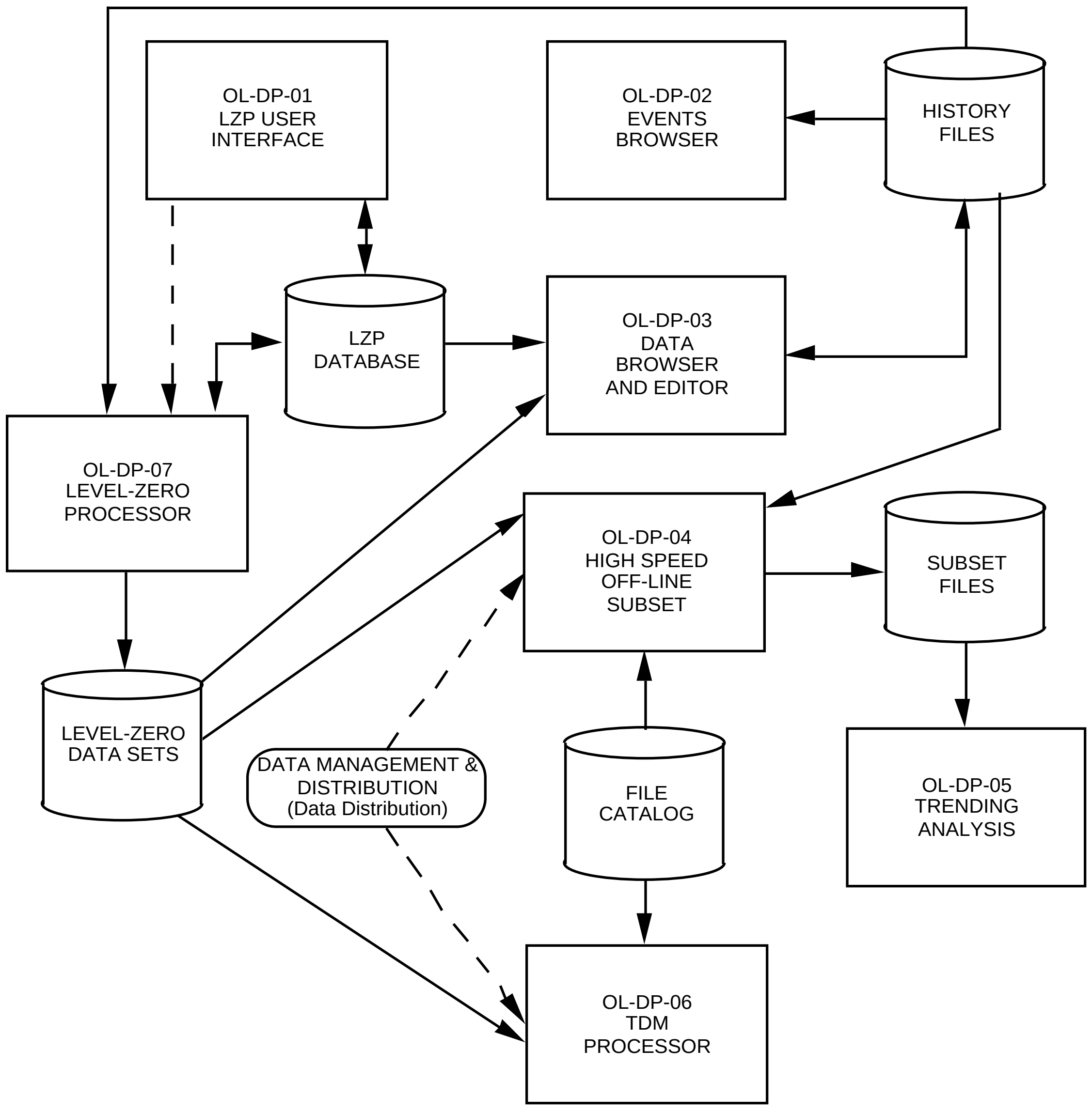


Figure 2.2-2 Off-Line Data Production and Analysis

2.2.3 Off-Line Mission Planning and Scheduling

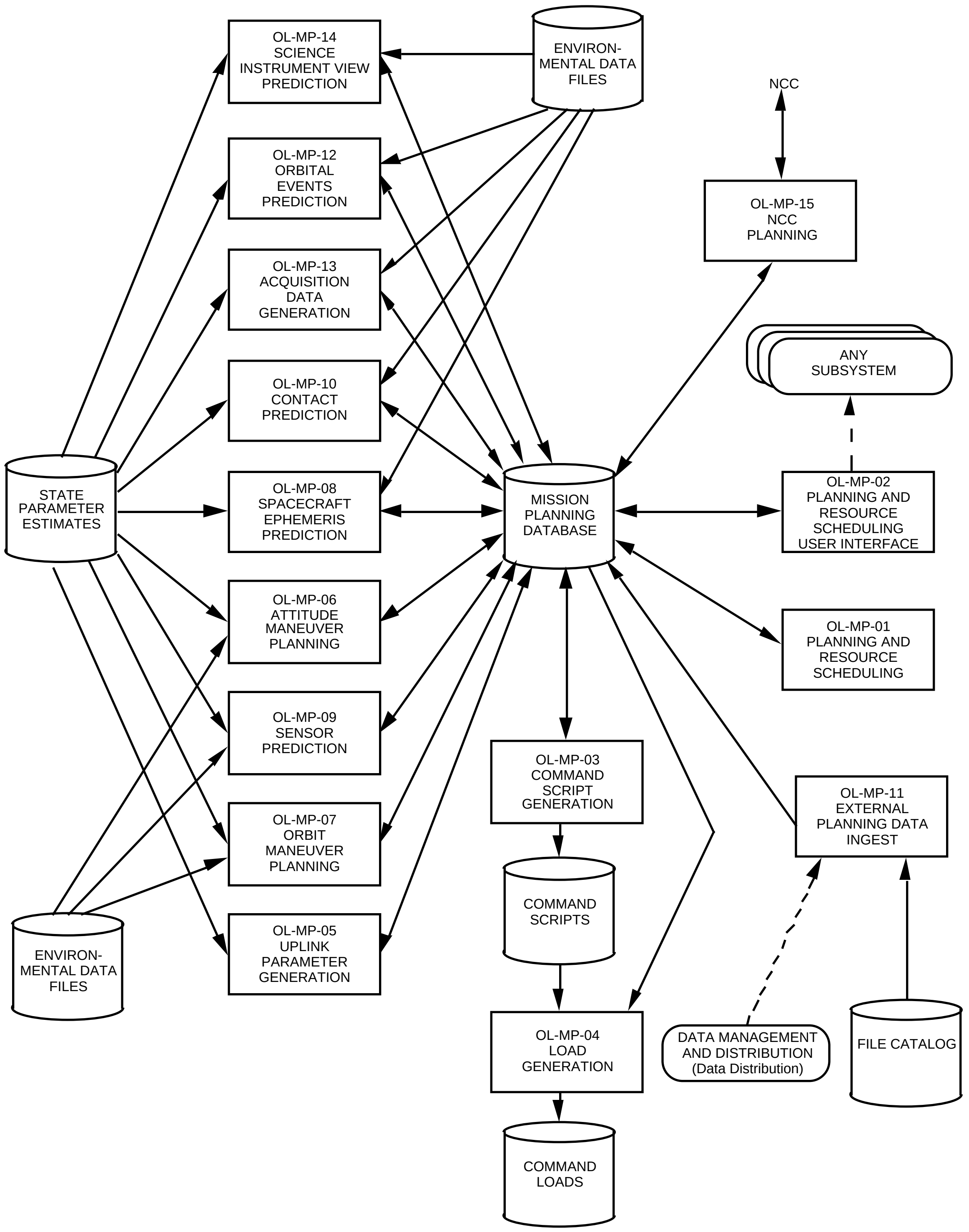
The Off-Line Mission Planning and Scheduling subsystem provides the information required for communications scheduling, maneuver support, and onboard spacecraft control. This subsystem includes various prediction capabilities, uplink table and command load generation, and planning and resource scheduling.

The functions within this subsystem are highly intertwined. For example, various predicted parameters (based on state parameter estimates from the Off-Line State Determination and Validation subsystem) are passed to the remaining functions in order to accomplish required control and scheduling support. The key to this interaction is a centralized mission planning database. This database may be accessed and updated using the Planning and Resource Scheduling User Interface. Changes to the database, via the user interface or other planning element, may trigger processing in appropriate elements to perform consistency checking and any other necessary processing.

The outputs from this subsystem are extracted from the mission planning database. These outputs are typically used by the mission operations team as well as external entities, such as the communications network, science operations center and centers of expertise. In addition, the Command Script Generation element uses scheduling and other information in the mission planning database to generate command scripts. The Load Generation element then uses these scripts to generate command loads. These loads are then passed to the Real-Time Command subsystem for uplink to the spacecraft.

NOTE

This architecture is based on the mission planning “data manager” architecture used for the legacy Mission Operations Planning and Scheduling System (MOPSS). As has been proposed for MOPSS, this architecture also integrates command script generation with mission planning, though the exact legacy for the Command Script Generation element has not yet been fully determined. Also note that while the Load Generation function is often largely mission-specific there is a great deal of underlying commonality with parts of the generic Real-Time Commanding element. Further, the availability of generic commanding software in general may influence spacecraft designers to create compatible command systems across missions.



OL-MP-14 SCIENCE INSTRUMENT VIEW PREDICTION
ENVIRON-MENTAL DATA FILES
NCC
OL-MP-12 ORBITAL EVENTS PREDICTION
OL-MP-15 NCC PLANNING
OL-MP-13 ACQUISITION DATA GENERATION
ANY SUBSYSTEM
OL-MP-10 CONTACT PREDICTION
STATE PARAMETER ESTIMATES
OL-MP-08 SPACECRAFT EPHEMERIS PREDICTION
MISSION PLANNING DATABASE
OL-MP-02 PLANNING AND RESOURCE SCHEDULING USER INTERFACE
OL-MP-06 ATTITUDE MANEUVER PLANNING
OL-MP-01 PLANNING AND RESOURCE SCHEDULING
OL-MP-09 SENSOR PREDICTION
OL-MP-03 COMMAND SCRIPT GENERATION
OL-MP-07 ORBIT MANEUVER PLANNING
OL-MP-11 EXTERNAL PLANNING DATA INGEST
COMMAND SCRIPTS
ENVIRON-MENTAL DATA FILES
OL-MP-05 UPLINK PARAMETER GENERATION
OL-MP-04 LOAD GENERATION
DATA MANAGEMENT AND DISTRIBUTION (Data Distribution)
FILE CATALOG
COMMAND LOADS

Figure 2.2-3 Off-Line Mission Planning and Scheduling

2.2.4 Off-Line Simulation

The Off-Line Simulation subsystem generates simulated spacecraft data. This subsystem employs high-fidelity models in order to predict spacecraft behavior for a user-specified set of conditions. The elements of this subsystem provide both history files and simulated telemetry streams containing the resulting spacecraft state parameters.

The data from this subsystem is passed to the Off-Line Mission Planning and State Determination and Validation subsystems for testing. mission planning, and operator training. The mission planning functions include, but are not limited to, maneuver design, thermal and power modeling, and attitude prediction. Operators are trained with this subsystem in all off-line telemetry and mission planning operations.

NOTE

The Off-Line Spacecraft Simulator is similar in function to current Flight Dynamics Division high-fidelity simulators, except it includes the modeling of non-flight-dynamics-related spacecraft systems. The Off-Line Telemetry simulator is similar in function to the current GTSIM system, except that it must also pack simulated telemetry data appropriately in simulated packets. It is intended that the off-line simulators should evolve to maximize software commonality at the subelement level with their Real-Time Simulation subsystem counterparts.

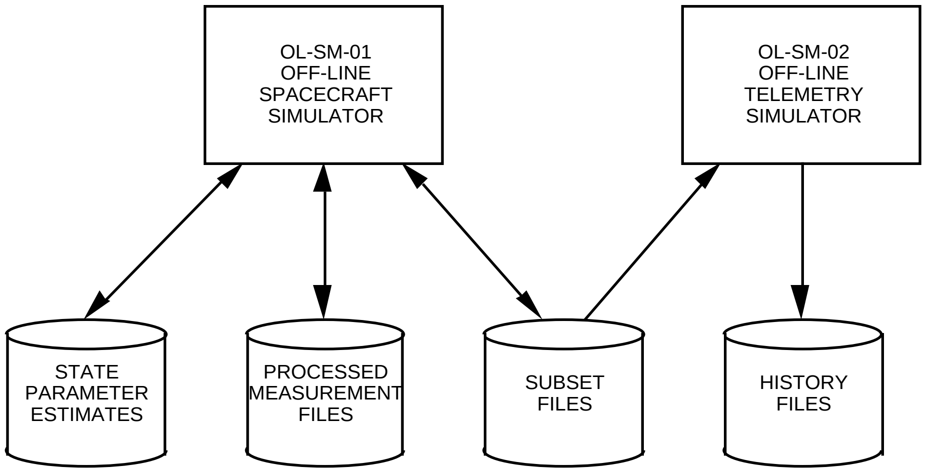


Figure 2.2-4 Off-Line Simulation

2.2.5 Off-Line State Determination and Validation

The Off-Line State Determination and Validation subsystem provides elements that determine the spacecraft flight dynamics and on-board systems state. Flight dynamics state determination consists of processing telemetry subset files (from either the Real-Time Telemetry subsystem or the Off-Line Data Production and Analysis subsystem) and then using these measurements to estimate the spacecraft attitude and attitude sensor alignment and calibration parameters. These state parameter estimates can be distributed as products as well as used to validate on-board state parameter computations and to monitor sensor degradation and many other phenomena. The state parameter estimates are also used by the Off-Line Mission Planning subsystem and by the Real-Time Attitude Determination element of the Real-Time Telemetry subsystem.

The Ground Reference Image Maintenance element is used to maintain and analyze the ground reference image (GRI) of the on-board systems state. This generally consists of an image of the OBC memory and the state of any other relevant on-board hardware. The GRI may be manually updated by the operator, updated by the Real-Time Commanding element (in the Real-Time Command subsystem) based on actual commands sent to the spacecraft, or predicted based on a planned command load from the Off-Line Mission Planning subsystem. The Ground Reference Image Maintenance element also allows the current or predicted GRI to be compared against a dump/image file generated by the Real-Time Telemetry subsystem in order to validate the operation of the spacecraft.

NOTE

Currently, the TPOCC effort and the Flight Dynamics Division are in the process of defining a special ASCII "attitude telemetry file" that is created by special TPOCC software that reads data from the data server. The Renaissance architecture defines the use of subset files, also used for other purposes, instead of a special-purpose attitude interface. Note also that it may be useful to give some additional consideration in the future to how the GRI and attitude state determination and validation functions might be more tightly combined. Orbit determination capabilities may also be added to this subsystem.

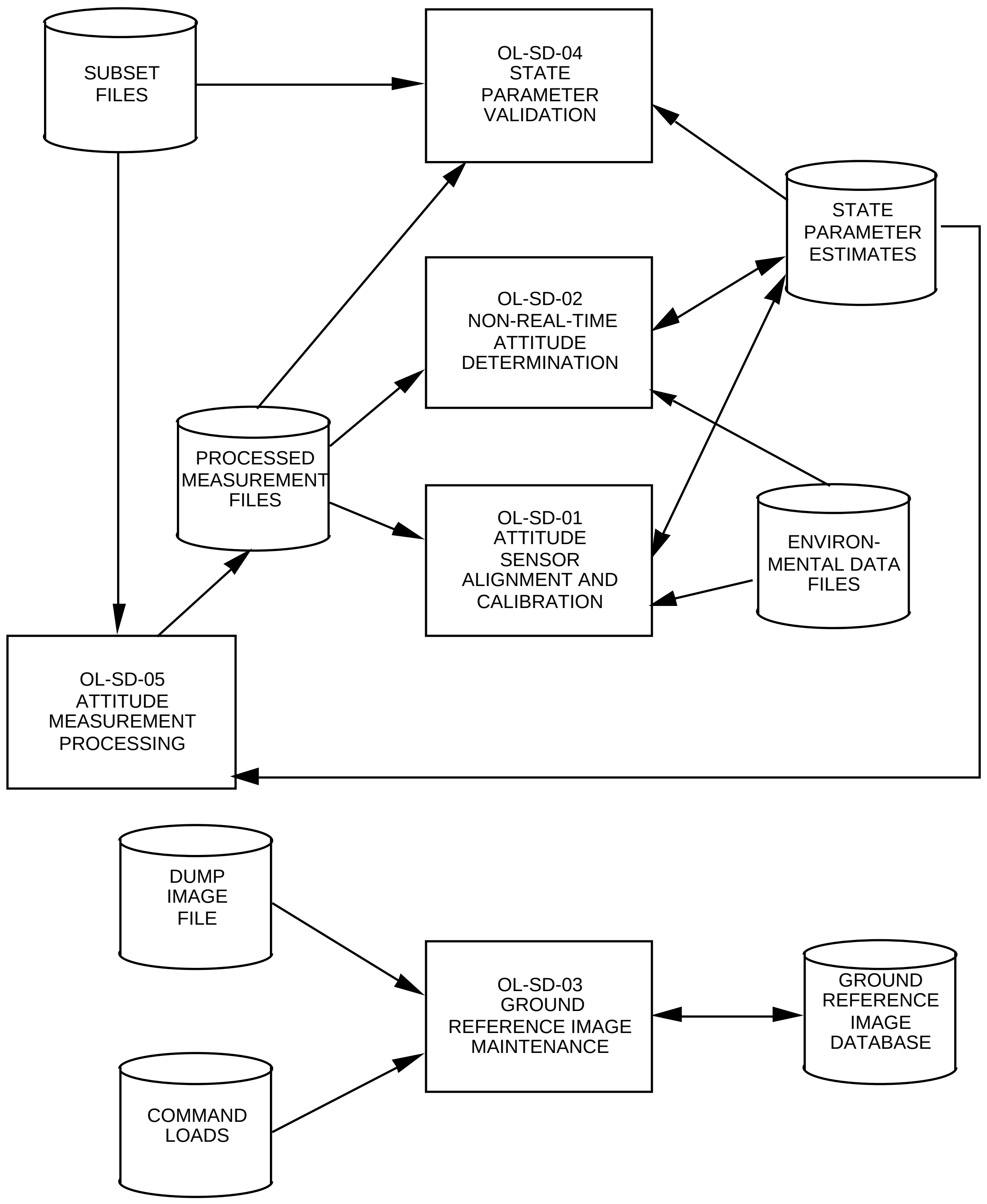

SUBSET FILES
OL-SD-04 STATE PARAMETER VALIDATION
STATE PARAMETER ESTIMATES
OL-SD-02 NON-REAL-TIME ATTITUDE DETERMINATION
PROCESSED MEASUREMENT FILES
OL-SD-01 ATTITUDE SENSOR ALIGNMENT AND CALIBRATION
ENVIRON-MENTAL DATA FILES
OL-SD-05 ATTITUDE MEASUREMENT PROCESSING
DUMP IMAGE FILE
OL-SD-03 GROUND REFERENCE IMAGE MAINTENANCE
GROUND REFERENCE IMAGE DATABASE
COMMAND LOADS

Figure 2.2-5 Off-Line State Determination and Validation

2.2.6 Off-Line User Interface Support

The Session Manager element provides a general capability for running, sequencing and monitoring other off-line elements as well as a parameter editor that allows the operator to browse and edit setup parameter files for other elements. This is the primary overall executive capability for the off-line segment. The Session Manager initiates and monitors other elements using operating system and/or distributed system APIs.

The Off-Line User Interface Support subsystem also provides the off-line capability to create custom user interface displays for use by the Real-Time Display element (in the Real-Time User Interface subsystem). The Display Builder element generates user interface language (UIL) files of display definitions that are then translated by the UIL Compiler into user interface definition (UID) files that can be used by the Real-Time Display element. (Both the Display Builder and the UIL Compiler are COTS products).

The STOL syntax checker is used to verify new STOL procedures prior to use in the real-time elements of the system.

NOTE

The Session Manager element is based on the session manager concept from the legacy Flight Dynamics Distributed System User Interface/Executive (UIX). Currently most off-line legacy applications have not been designed to operate with the UIX session manager (or, in particular, to use the parameter files it creates), so some adaptation will be necessary (of the UIX session manager and/or other legacy applications). Also, note that the display building capabilities of this subsystem are currently only used for the Real-Time User Interface. Legacy off-line systems currently use a wide range of user interface approaches. It is planned to eventually evolve these into a more common approach, in which case it may be possible to generalize the user interface building capabilities of this subsystem.

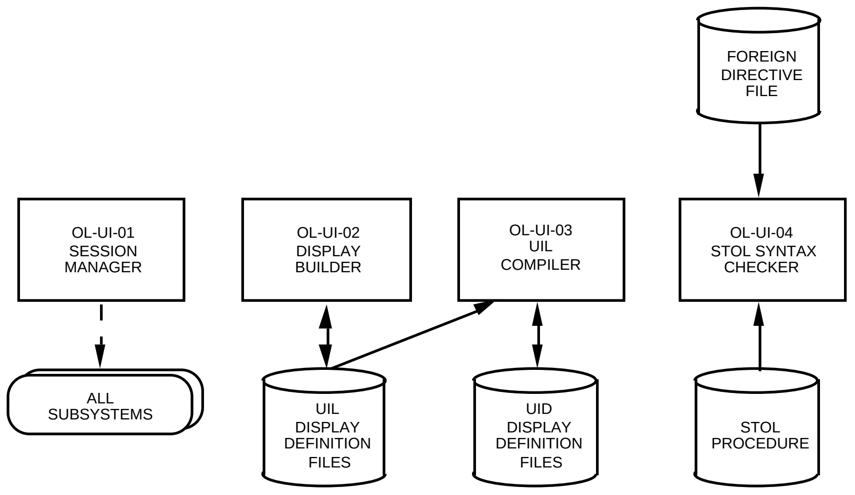


Figure 2.2-6 Off-Line User Interface Support

2.3 Foundation Subsystems

Much of the foundation software is system software based on the various open-system standards discussed in the *Renaissance Generic Architecture* (reference TBD) and *Renaissance Standards* (Stottlemyer, 1994) documents. The software supporting these basic system capabilities is not included in this catalog. However, there are certain general capabilities that are built on top of this lower-level foundation but which are not specific to either the real-time or off-line architecture segments. The software for these capabilities is grouped into four subsystems discussed in the following paragraphs and the corresponding elements are included in the catalog in Section 3. All the elements in the foundation subsystems are COTS software and the capabilities they provide is generally available to all other (real-time and off-line) subsystems.

2.3.1 Electronic Mail

The Electronic Mail (EMAIL) Engine provides a general capability for creating, distributing and receiving electronic messages and also sending FAXs via the FAX Server. The EMAIL Agent provides access to this capability for user of the ground data system. However, applications can also directly access the EMAIL Engine in order to use the electronic mail capability for the automated sending and receipt of data.

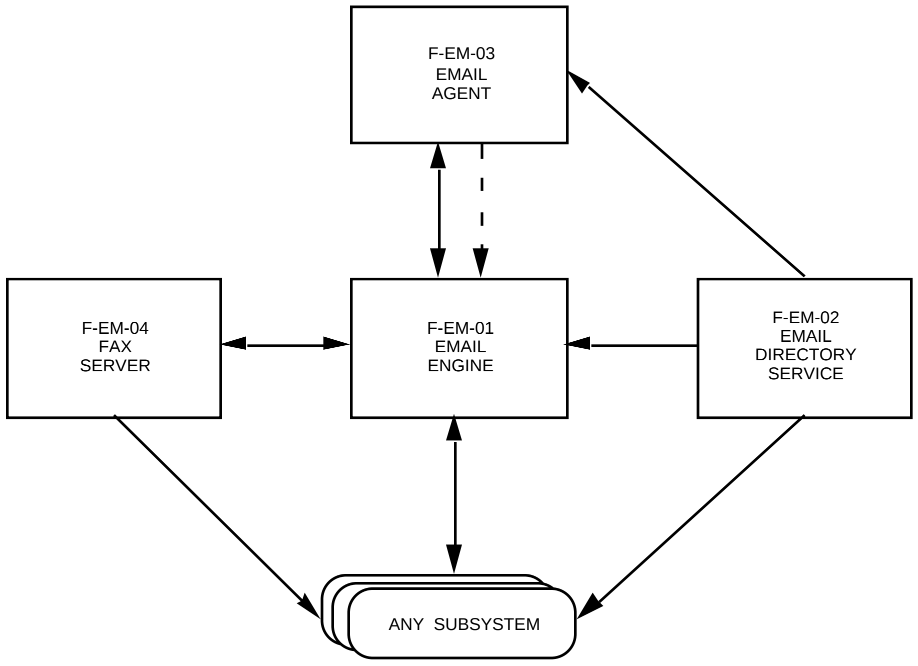


Figure 2.3-1 Electronic Mail

2.3.2 Network Management

The Network Management System provides an umbrella for monitoring, management and control of networking and communications components, applications, file servers and workstations in the ground data system. It gathers information from Network Management Agents that monitor network hardware components and Remote Monitoring Agents that monitor remote applications. These agents contain network management information bases that collect the information needed by the Network Management System.

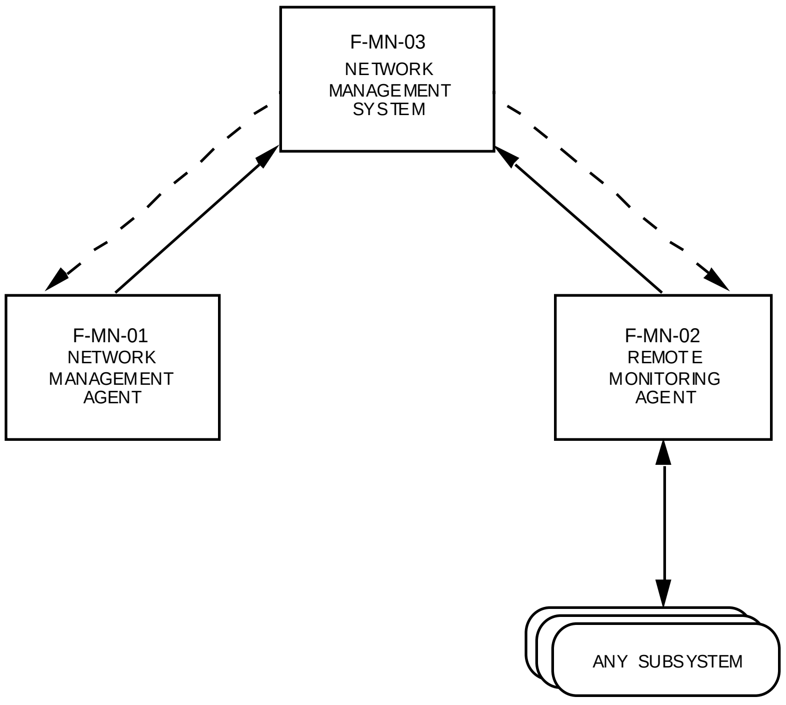


Figure 2.3-2 Network Management

2.3.3 Network Multicast

The Network Multicast subsystem consists of a single IP Multicast element. This element can receive an IP data stream from a source element and distribute it to multiple IP destinations. This eliminates the normal point-to-point limitation of IP addressing and, in particular, allows data streams from external sent to a single IP address in the MOC to be distributed to multiple destinations within the ground data system.

2.3.4 Network Security Services

The Network Security Services subsystem consists of a single Security Server element. This element provides access control to and from internal ground data system platforms and services as well as authentication of users.

2.3.5 Network Time Service

The Network Time Service subsystem provides the mechanism for establishing a consistent time reference across the various networked ground data system components. The subsystem can connect to an external time reference (such as NASA-36) via the Network Time Source element. This time reference is then distributed by the Network Time Server to Network Time Clients across the network. These clients then can provide a consistent time reference to applications (generally via setting the system clock of the platform on which the agent resides).

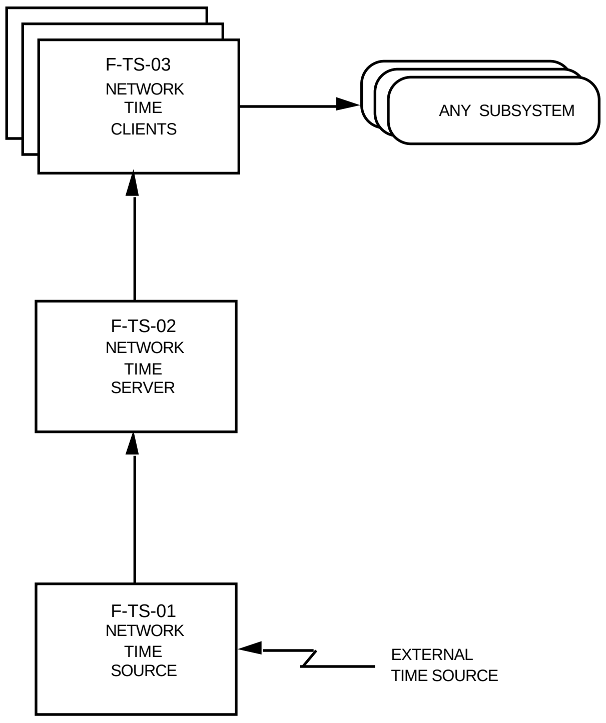


Figure 2.3-3 Network Time Service

3. Building Block Descriptions

This section provides a summary listing and catalog of the current set of Renaissance building blocks. Section 3.1 provides a summary list of all element-level building blocks. Some of the element-level building blocks are constructed and tailored using subelement components. Section 3.2 lists these lower level subelement building blocks. Finally, Section 3.3 provides the complete building block catalog.

3.1 Element Summary List

This subsection provides a summary listing of the element-level building blocks identified to date. The elements are sorted by either the real-time, off-line, or foundation architectural segments. An identifier, name, and functional description are provided for each element. The identifier has been selected to allow users to quickly recognize elements according to various classifications. The first field in the identifier shows whether the element belongs to the real-time, off-line, or foundation categories of subsystems; the second field shows the actual subsystem to which the element belongs. The first two fields in the identifier have been chosen in a manner such that users may use them as mnemonics for quick reference. Table 3-1 lists the identifier fields and their definitions. Table 3-2 is the actual summary list.

Identifier	Definition
RT	Real-Time
OL	Off-Line
F	Foundation
CM	Command
DS	Distribution
EX	External Interface
HS	History
SM	Simulation
TM	Telemetry
US	User Interface
DM	Data Management and Distribution
DP	Data Production and Analysis
MP	Mission Planning
SD	State Determination and Validation
UI	User Interface
EM	Electronic Mail
MN	Network Management
ML	Network Multicast
SS	Network Security Services
TS	Network Time Service

Table 3-1: Identifier Field Definitions

File Name : R1.EPS
Title : TBL3-2A.RSL
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

File Name : R2.EPS
Title : TBL3-2A.RSL
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

File Name : R3.EPS
Title : TBL3-2A.RSL
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

File Name : R4.EPS
Title : TBL3-2A.RSL
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

File Name : R5.EPS
Title : TBL3-2A.RSL
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

File Name : P1.EPS
Title : Const
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

File Name : P2.EPS
Title : Const
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

File Name : P3.EPS
Title : Const
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

File Name : P4.EPS
Title : Const
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

File Name : P5.EPS
Title : Const
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

File Name : P6.EPS
Title : Const
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

ile Name : P7.EPS
tle : Const
reator : Windows PSCRIPT
ages : 0

Table 3-2: Summary List of Elements

File Name : F1.EPS
Title : TBL3-2A.RSL
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

File Name : F2.EPS
Title : TBL3-2A.RSL
Creator : Windows PSCRIPT
Pages : 0

Table 3-2: Summary List of Elements

3.2 Subelement Summary List (TBS)

3.3 Building Block Descriptions

This subsection provides detailed descriptions for each of the building blocks identified to date. The following information is provided for each building block description:

<u>Functional Information</u>

Element Title:	Title of building block (for this document version, all building blocks are elements)
Element ID:	A unique identifier
Date:	Date the description page was last updated
Subsystem:	The subsystem to which the element belongs
Functional Description:	Overview of building block functionality
Functional Capabilities:	List of building block functional capabilities
Performance Profile:	Building block performance requirements
Issues/Comments:	Self-explanatory

<u>Interface Information</u>

Interface Data:	Data passed by interface
Source:	Element that transmits data to this element
Destination:	Element that receives data transmitted by this element
Interface Name:	Self-explanatory
Interface API:	Application Program Interface for this interface
Client/Server:	Designates element interface as client or server

<u>Implementation Information</u>

Provider Name:	Name of organization providing this building block implementation
Platforms:	Platforms on which this building block implementation is known to run
Availability Date:	Date that this building block implementation is available

Characteristics:	Implementation-specific characteristics of this building block

Table 3-3 contains a detailed description of each building block.

Appendix A - Glossary

Building Block: A loose term for generic or tailored subsystem, element or subelement.

Data Parameters: Data maintained by a server or producer element that may be provided on to client or consumer element.

Generic: A *generic* system, subsystem, element or subelement is one that is always used with no modifications from mission to mission. Mission configurability may be achieved through data parameters or through *tailoring*.

Mission-Specific: A *mission-specific* system, subsystem, element or subelement is one that is developed and managed for a specific mission.

Software Backplane: The set of Renaissance-standard APIs for providing communications between software elements. Conceptually, elements "plug into" the standard interfaces of the backplane (in analogy to a hardware backplane).

Software Element: An independently executing software unit that communicates with other software elements solely through the standard interfaces defined by the Renaissance software backplane. A software element is constructed from subelements that may be linked together or may be separate tasks or processes. Subelements within a software element may communicate among themselves by means other than the backplane.

Software Subelement: A configuration-controlled component of one or more software elements. Subelements may themselves have structure, but the subelement level is the lowest level of concern for configuration of a system for a particular mission.

Subsystem: A standard decomposition of a software system, encompassing those software elements necessary to carry out a specific high-level functional area. For example, Spacecraft Operations Monitoring and Mission Planning might be subsystems (these examples are for illustrative purposes only, and do not imply a definitive decomposition of Renaissance systems).

System: A complete operational environment with clear boundaries and well-defined external interfaces crossing these boundaries, such as the Mission Operations Center (MOC), Mission Science Center (MSC) and spacecraft Integration and Test (I&T) systems.

Tailored: A *tailored* subsystem, element or subelement is one with some mission-specific code added to a generic part in some standard way (e.g., linked in via well-defined hooks), or one in which there is a mission-specific selection from alternatives in creating the element or subelement. This may include selection of a COTS product as a part of the element or subelement.

Appendix B - Acronym List

ADPE	Automated Data Processing Equipment
API	Application Program Interface
CDE	Common Desktop Environment
COE	Center of Expertise
COTS	Commercial Off The Shelf
DBMS	Data Base Management System
DCE	Distributed Computing Environment
Email	Electronic Mail
FDF	Flight Dynamics Facility
FTP	File Transfer Protocol
GDS	Ground Data System
GN	Ground Network
I&T	Integration and Test
ICD	Interface Control Document
IPC	Inter-Process Communication
Kbps	Kilobits per second
LAN	Local Area Network
LZP	Level Zero Process
MOC	Mission Operations Center
MOT	Mission Operations Team
NMS	Network Management System
Nascom	NASA Communications
PACOR	Packet Processor
PDB	Project Data Base
PICS	PACOR II Information and Control System
POCC	Payload Operations Control Center
POSIX	Portable Open System Interconnect Executive
QAWS	Quality Analaysis Work Station
RF	Radio Frequency
RPC	Remote Process Call
RT	Real-time
RTADS	Real Time Attitude Determination System
SPEC 1170	Profile of standards developed by X/Open
TCSEC	Trusted Computer Security Evaluation Criteria
TDM	Time Division Multiplexed (telemetry)
UDP	User Datagram Protocol (IP connectionless protocol)
UI	User Interface
UIX	User Interface and executive
UTC	Coordinated Universal Time

VC	Virtual Channel
XDR	External Data Representation (SUN utilities)
XPG	X/Open Portability Guide

References

Green, D., et al., *Flight Dynamics Distributed Systems (FDDS) System Architecture*, Goddard Space Flight Center, Flight Dynamics Division, 552-FDD-94/036R0UD0, August 1994

Principe, C., et al., *Packet Processor II (Pacor II) Detailed Design Specification*, Volume 1: System Design Overview, Goddard Space Flight Center, Information Processing Division, 560-7DDD/0193, May 1993

Renaissance Team, *Renaissance Configuration Guide*, Goddard Space Flight Center, Mission Operations and Data Systems Directorate, to be written

Renaissance Team, *Renaissance Generic Architecture*, Goddard Space Flight Center, Mission Operations and Data Systems Directorate, to be written

Schwarz, B., *Transportable Operations Control Center (TPOCC) Implementation Guide for Release 9*, Goddard Space Flight Center, Mission Operations Division, 511-4SSD/0393, February 1993

Stottlemyer, A., *Renaissance Standards (Draft)*, Goddard Space Flight Center, Mission Operations and Data Systems Directorate, August 1994

Zhu, C., et al., *Transportable Payload Operations Control Center (TPOCC) Detailed Design Specification for Release 10*, Goddard Space Flight Center, Mission Operations Division, 511-4DDS/0693, February 1994